
CHAPTER# 8

Functional Dependencies & Normalization

Relvar DCL:

D#	C#	LOCATION	QUANTITY	STATUS
D1	C1	SHELTON	30	40
D1	C2	SHELTON	20	40
D1	C3	SHELTON	40	40
D1	C4	SHELTON	20	40
D1	C5	SHELTON	10	40
D1	C6	SHELTON	10	40
D1	C7	SHELTON	25	40
D1	C8	SHELTON	15	40
D2	C1	BRIDGEPORT	30	45
D2	C2	BRIDGEPORT	40	45
D3	C2	SHELTON	20	40
D4	C2	STAMFORD	20	50
D4	C4	STAMFORD	30	50
D4	C5	STAMFORD	40	50

-
- Functional dependency: a many-to-one relationship from one set of attributes to another within a given relvar.
 - Example: for the DEALER relvar DCL, there is a functional dependency from the set of attributes {D#, C#} to the attribute QUANTITY
 - For any given value for the pairs D# and C#, there is only just one corresponding value of attribute QUANTITY
 - Many distinct values of the pair of attributes D# and C# can have the same corresponding value for attribute QUANTITY

Definitions

- Cases:
 - the value of a relvar at any given point of time
 - the set of all possible values the relvar might assume at different times

- Case A:

Let r be a relation, and let X and Y be arbitrary subsets of the set of attributes of r . Then we say that Y is functionally dependent on X ($X \rightarrow Y$), *iff* each X value in r has associated with precisely one Y value in r .

- Example:

{ D# } → { LOCATION }

{ D#, C# } → { LOCATION }

{ D#, C# } → { LOCATION, QUANTITY }

{ D#, C# } → { D# }

{ D#, C# } → { D# , C# , LOCATION }

-
- Determinant: left hand side of the FD
 - Dependent: right hand side of the FD
 - When the set of attributes contains one set we drop the set braces: $D\# \rightarrow \text{LOCATION}$

-
- Case B: FDs that hold for all possible values of the relvar
 - $D\# \rightarrow \text{LOCATION}$
 - It means that a given Dealer has precisely one corresponding city.
 - FDs holding all time means that it is an integrity constraint

-
- Definition of Case B:
 - Let R be a relation variable, and let X and Y be arbitrary subsets of the set of attributes of R . Then X is functionally dependent on Y iff in every possible legal value of R , each X value has associated with precisely one Y value.

- Example:

- { D# } → { LOCATION }
- { D#, C# } → { LOCATION }
- { D#, C# } → { LOCATION , QUANTITY }
- { D#, C# } → { D# }
- { D#, C# } → { D# , C# ,
LOCATION,QUANTITY }

- These two are not true:

- { D# } → { QUANTITY }
- { QUANTITY } → { D# }

-
- If relvar R satisfies the FD $A \rightarrow B$ and A is not a candidate key, then R will involve some kind of redundancy.
 - Example: $D\# \rightarrow \text{LOCATION}$
 - notice that a given DEALER in a given LOCATION appears many times

-
- The complete set of FDs for a given relvar can be very large
 - It is desirable to reduce the set of FDs to a minimal number
 - FDS are integrity constraints that the DBMS needs to enforce

Trivial and Nontrivial Dependencies

- A dependency is trivial if it cannot possibly not be satisfied

$$\{ D\#, C\# \} \rightarrow D\#$$

- FD is trivial IFF the right-hand side is a subset of the left-hand side

Normalization

Normalization(1NF, 2NF, 3NF, BCNF)

- Example
 - D { D# , DNAME, STATUS, LOCATION
Primary Key { D# }
 - C { C#, MODEL , YEAR, LOCATION }
Primary Key { C# }
 - DC { D# , C# , QUANTITY }
Primary Key { D# , C# }
Foreign Key { D# } References D
Foreign Key { C# } References C

-
- Why this design is correct?
 - MODEL in C ?
 - Quantity in DC?
 - Suppose the DEALER LOCATION attribute moved to DC relvar (shown as DCL relvar)
 - New relvar DCL has redundancy
 - A good design principle is “one fact in one place”

Relvar

DCL:

D#	C#	QUANTITY	LOCATION	STATUS
D1	C1	30	SHELTON	40
D1	C2	20	SHELTON	40
D1	C3	40	SHELTON	40
D1	C4	20	SHELTON	40
D1	C5	10	SHELTON	40
D1	C6	10	SHELTON	40
D1	C7	25	SHELTON	40
D1	C8	15	SHELTON	40
D2	C1	30	BRIDGEPORT	45
D2	C2	40	BRIDGEPORT	45
D3	C2	20	SHELTON	40
D4	C2	20	STAMFORD	50
D4	C4	30	STAMFORD	50
D4	C5	40	STAMFORD	50

-
- Properties of Relations
 - There are no duplicate tuples
 - Tuples are unordered
 - Attributes are unordered
 - Each tuple contains exactly one value for each attribute
 - First Normal Form (1NF) (fourth property)
 - All relations are normalized according to the definition of 1NF.

Further Normalization

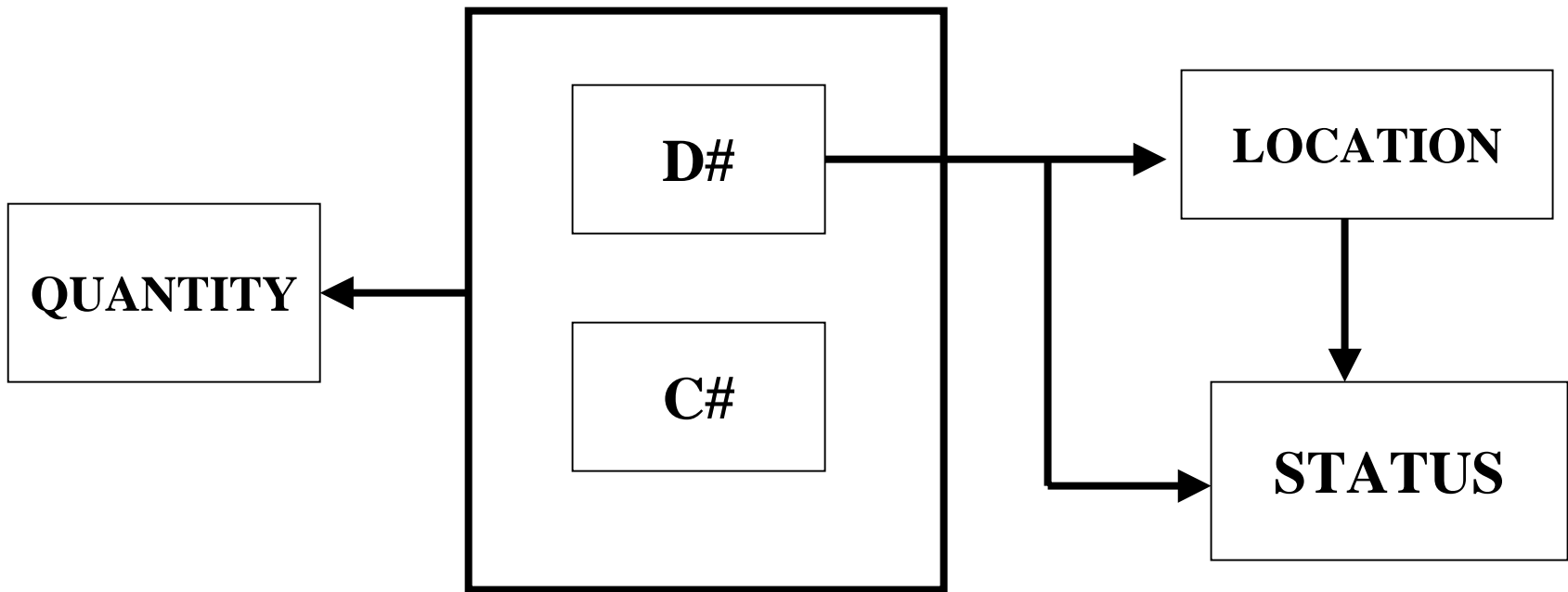
- It is desirable to replace relvars such as DCL with other relvars that don't contain redundant information
- This process is called Normalization
- A relvar that satisfies a certain set of conditions is said to be in a particular normal form.

Normal Forms

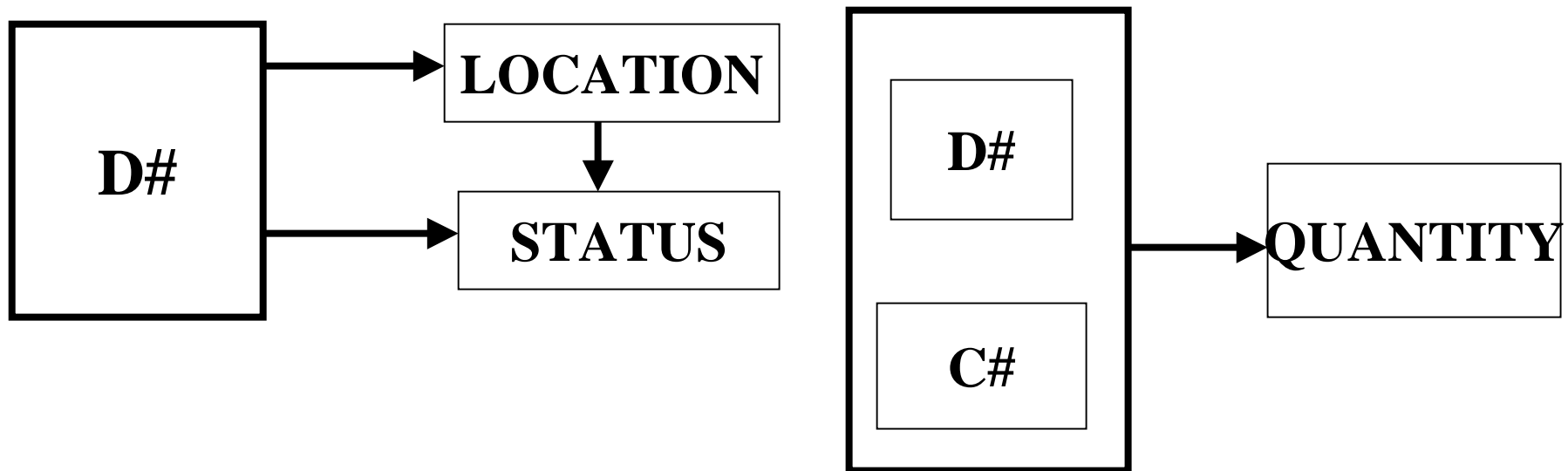
- The first three normal forms are called 1NF, 2NF, 3NF introduced by Codd
- The higher the degree the lower the redundancy (repetition of facts)
- Database designer should target 3NF or higher
- Normalization Procedure:
 - A relvar in some given normal form can be replaced by a set of relvars in some more desirable form

Definitions of the normal forms:

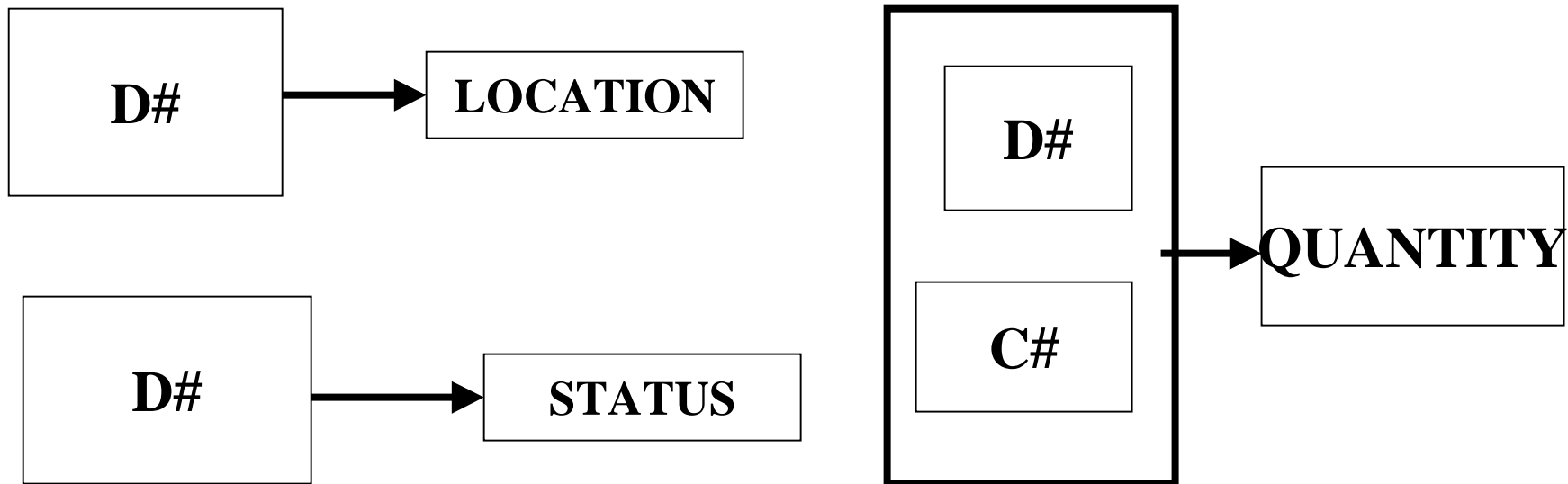
- **1NF:** A relvar is in first normal form if and only if, in every legal value of that relvar, every tuple contains exactly one value for each attribute.



-
- **2NF:** A relvar is in 2NF if and only if it is in 1NF and every non-key attribute is irreducibly dependent on the primary key



-
- **3NF:** A relvar is in 3NF if and only if the non-key attributes are
 1. Mutually independent and
 2. Irreducibly dependent on the primary key.



1NF Relvars

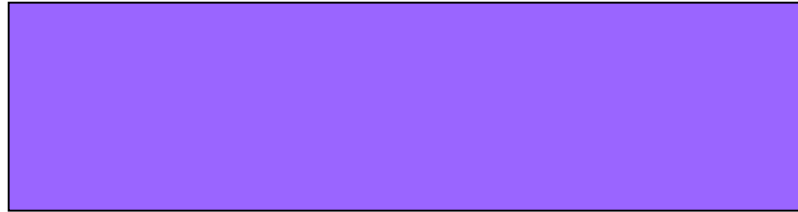
2NF Relvars

3NF Relvars

BCNF Relvars

4NF Relvars

5NF Relvars



-
- The normalization procedure is reversible: normalization process is information preserving while eliminating redundancy
 - Boyce/Codd normal form (BCNF) is a modification of the original 3NF introduced by Codd
 - Fagin defined 4NF and 5NF

Non-loss Decomposition and Functional Dependencies

- The normalization process involves breaking (decomposing) a given relvar into other relvars
- The procedure has to be reversible so that no information is lost
- We are considering only non-loss decompositions
- Example:
 - Case (a) non-loss decomposition
 - Case (b) lossy decomposition

D:

D#	STATUS	LOCATION
D1	40	BRIDGEPORT
D2	20	STAMFORD
D3	40	SHELTON

CASE:A

DS

D#	STATUS
D1	40
D2	20
D3	40

DL

D#	LOCATION
D1	BRIDGEPORT
D2	STAMFORD
D3	SHELTON

CASE:B

DS

D#	STATUS
D1	40
D2	20
D3	40

SL

STATUS	LOCATION
40	BRIDGEPORT
20	STAMFORD
40	SHELTON

-
- Process of decomposition is a process of projection: DS,DL,SL are projections of D
 - Joining DS and DL produces D
 - Joining DS and SL doesn't produce D
 - Reversibility means that the original relvar is equal to the join of its projections
 - Decomposition is done by projection
 - Recomposition is done by joining

-
- What conditions should be satisfied by the projections of the relvar to guarantee that joining the projections will produce the original relation?
 - D satisfies: $D\# \rightarrow \text{LOCATION}$
 $D\# \rightarrow \text{STATUS}$
 - Heath's theorem:
Let R {A,B,C} be a relvar, where A, B, and C are sets of attributes. If R satisfies the FD $A \rightarrow B$, then R is equal to the join of its projections on {A,B} and {A,C}

More on Functional Dependencies

- Left-irreducible FDs:
 - LOCATION is functionally dependent on D# only
 - {D#, C# } \rightarrow LOCATION : not left irreducible
 - D# \rightarrow LOCATION : is left irreducible
- Every arrow is an arrow out of a candidate key
 - If there are any other arrows that difficulties arise
 - Normalization is a procedure for eliminating arrows that are not arrows out of candidate keys

-
- FDs are semantic notion: $D\# \rightarrow \text{LOCATION}$
 - If there is a real constraint that each supplier is located in precisely one city
 - That constraint must be observed in the database
 - It must be specified in the database definition
 - The way to specify is to declare it as an FD
 - Normalization leads to a very simple way of declaring FDs

Update anomalies with 1NF

- Example: DEALER-LOCATION redundancy
(D# \rightarrow LOCATION)
 - Insert: we can't insert the fact that a DEALER is located in a particular LOCATION until DEALER has at least one car(C#)
(D7,NEWYORK)
 - Delete: If we delete the tuple with value D# equal D3 and C# equal C2, we lose information that D3 is located in SHELTON.

-
- Update: The LOCATION value for a given DELAER appears in DCL many times.
 - Whenever a LOCATION need to be updated the whole relation has to be searched for updates.
 - Solution:
 - Second { D# , Location, Status }
 - SP {D# ,C# , Quantity }
 - All the previous problems with DCL were eliminated

SECOND

D#	LOCATION	STATUS
D1	SHELTON	40
D2	BRIDGEPORT	45
D2	BRIDGEPORT	45
D3	SHELTON	40
D4	STAMFORD	50

SP

D#	C#	QUANTITY
D1	C1	30
D1	C2	20
D1	C3	40
D1	C4	20
D1	C5	10
D1	C6	10
D1	C7	25
D1	C8	15
D2	C1	30
D2	C2	40
D3	C2	20
D4	C2	20
D4	C4	30
D4	C5	40

Procedure to convert 1NF to 2NF

- R { A, B, C, D }
Primary key { A, B }
/ assume $A \rightarrow D$ holds */*
- R can be replaced by two projections R1 and R2 as follows:
R1 { A, D }
Primary Key { A }
R2 { A, B, C }
Primary key { A, B }
Foreign Key { A } References R1

Update anomalies in 2NF

- Example: Second relvar, LOCATION → STATUS
 - Insert: we can't insert a LOCATION status unless we have a DEALER in that LOCATION.
 - Delete: If we delete the sole tuple for a particular LOCATION, we delete the DEALER information and LOCATION STATUS.
 - Update: The STATUS of a given LOCATION appears many times
 - Solution: replace Second with 2 3NF relvars
 - DL {D # , LOCATION } CS {LOCATION, STATUS }

Procedure to convert 2NF to 3NF

- R { A, B, C }
Primary key { A }
/ assume $B \rightarrow C$ holds */*
- R can be replaced by two projections R1 and R2 as follows:
R1 { B , C }
Primary Key { B }
R2 { A, B }
Primary key { A }
Foreign Key { B } References R1

Dependency Preservation

- Decomposition has to be non-loss
- Example: Second relvar
 - It has the following FDs: $D\# \rightarrow \text{LOCATION}$,
 $\text{LOCATION} \rightarrow \text{STATUS}$
 - By transitivity: $D\# \rightarrow \text{STATUS}$
 - One possible decomposition, called A:
 - DL { $D\#$, LOCATION }
 - DS { $D\#$, STATUS }
 - Another possible decomposition, called B:
 - DL { $D\#$, LOCATION }
 - LC { LOCATION, STATUS }

-
- Both decompositions are non-loss
 - Both projections are in 3NF
 - Is it possible to insert the information that a particular LOCATION has a particular STATUS unless the same DEALER is in the same LOCATION ?

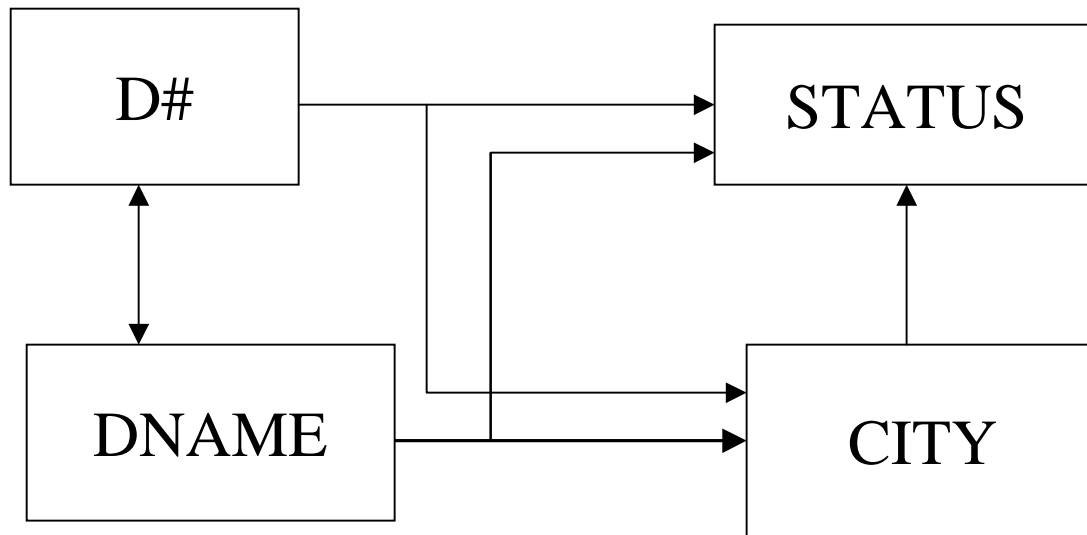
-
- Projections in B are independent: updates can be made to either without regard to the other as long as referential integrity constraints hold
 - In decomposition A, updates have to be monitored to ensure $LOCATION \rightarrow STATUS$
 - The projections in A are dependent
 - Projections in A are database constraint that spans two relvars and will be maintained by procedural code.
 - All dependencies in A are maintained by keys

-
- Projections R1 and R2 of a relvar R are independent IFF:
 - Every FD in R is a logical consequence of those in R1 and R2
 - The common attributes of R1 and R2 form a candidate key for at least one of the pair

BOYCE/CODD Normal Form (BCNF)

- Codd 3NF didn't deal with the case of a relvar:
 - had two or more candidate keys, such that the candidate keys were composite, and they overlapped
- BCNF is more stronger normalization than 3NF
- If the conditions mentioned above are not applicable, then 3NF and BCNF are equivalent

-
- Terminology
 - Determinant: left-hand side of an FD
 - Trivial FD: Left-hand side is a superset of right-hand side
 - BCNF: a relvar is in BCNF *iff* every nontrivial, left irreducible FD has a candidate key as its determinant
 - BCNF (less formally): A relvar is in BCNF *iff* the only determinants are candidate keys
 - BCNF (in other words): the only arrows in the FD diagram are arrows out of candidate keys



-
- D#,DNAME are candidate keys
 - CITY is determinant but not candidate key
 - This relvar is not in BCNF
 - Replace with R1 {D#,DNAME,CITY} and R2{CITY,STATUS}
 - R1 & R2 are in BCNF

Summary on Normalization

- The level of normalization is a matter of semantics
- It doesn't depend only on the value that the relvar has at a given point of time
- It is important to understand the meaning of the data (dependencies) before judging the a relvar is in 3NF
- It is not possible to prove by examining the relvar value that it is in 3NF
- The best to be done is to show that given values doesn't violate any of the dependencies

IN YOUR PROJECT YOU SHOULD TARGET BCNF

IMPORTANT

**In Your Database
Course Project
you must normalize
using BCNF**